(12)

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A162 668

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: TR 85 - 12 - 02 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Automated Generation of Microcontrollers	5. TYPE OF REPORT & PERIOD COVERED: Technical, interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Barry W. Jinks , David L. Pulfrey, Warren S. Snyder	8. CONTRACT OR GRANT NUMBER(s): MDA903-85-K0072 ARPA-4563, #2, code 5D30.
9. PERFORMING ORGANIZATION NAME AND ADDRESS: UW/NW VLSI Consortium, Computer Science Department, FR-35, University of Washington, Seattle, WA 98195	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: DARPA - IPTO, 1400 Wilson Boulevard, Arlington, Virginia 22209	12. REPORT DATE: December 1985
	13. NUMBER OF PAGES: 4
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): ONR, University of Washington, 315 University District Building, 1107 NE 45th St., JD-16, Seattle, WA 98195	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

microcontrollers, CFL, MAGIC, RNL, NETLIST, BILBO, RAM, PLA, ALU, RISC, CMOS, NORA

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The concept of an algorithmic microcontroller has been investigated. Software has been created which, when supplied with design parametrics, will generate several representations of the desire instance. This paper discusses the methodology followed during generator creation and the architecture and instruction set of the resulting family of controllers.

DTIC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Automated Generation of Microcontrollers

Barry W. Jinks* — David L. Pulfrey** — Warren S. Snyder***

Department of Computer Science
Seattle, Washington 98195
Technical Report 85-12-02
December, 1985

* Microtel Pacific Research Liaison, UW/NW VLSI Consortium, Seattle, WA98195
* Dept. of Electrical Engineering, U.B.C., Vancouver, B.C. V6T 1W5
** GTE Laboratories, 40 Sylvan Road, Waltham, MA 02254

ABSTRACT
The concept of an algorithmic microcontroller has been investigated. Software has been created which, when supplied with design parametrics, will generate several representations of the desired instance. This paper discusses the methodology followed during generator creation and the architecture and instruction set of the resulting family of controllers.

Funded in part by the Defense Advanced Research Projects Agency under Contract MDA903-85-K-0072.

Automated Generation of Microcontrollers

Barry W. Jinks* — David L. Pulfrey** — Warren S. Snyder***

* Microtel Pacific Research Liaison, UW/NW VLSI Consortium, Seattle, WA98195
** Dept. of Electrical Engineering, U.B.C., Vancouver, B.C. V6T 1W5
*** GTE Laboratories, 40 Sylvan Road, Waltham, MA 02254

ABSTRACT—The concept of an algorithmic microcontroller has been investigated. Software has been created which, when supplied with design parametrics, will generate several representations of the desired instance. This paper discusses the methodology followed during generator creation and the architecture and instruction set of the resulting family of controllers.

I. INTRODUCTION

The use of microcomputers as on-chip building-blocks is an attractive proposition for designers wishing to realize large integrated systems in silicon. Various approaches to this end are being explored, including standard library macrocells, silicon compilers and microcomputer generators. In a recent embodiment of the latter approach[1], the architecture of the microcomputer macrocells is essentially fixed but the user has control over important parameters such as data width, numbers of registers and the memory content and size. In the present work we have taken this concept a step further along the road to flexibility and area minimization by adopting a fully parametric approach to the design of a microcontroller.

By focusing on the microcontroller, rather than on a general purpose microcomputer, we have been able to limit the global instruction set and hence reduce the complexity of the generator design. The microcontroller generator is a software design environment which consists of a suite of subprograms capable of producing a number of different data base representations of a given instance. When furnished with the final system parametrics, the system synthesizes the mask geometries for a microcontroller to be instantiated in a user-specified system.

The microcontroller comprises a microprocessor, memory, communications protocol hardware and analog and digital I/O. It is intended to form a complete control system on a chip for use in telecommunications applications. The present work focuses on the microprocessor and memory portions of this generator.

II. GENERATOR DEVELOPMENT METHODOLOGY

Initial system design and functional level simulation was performed using a behavioral level simulator, after which a LISP-like description of the circuit, including parasitics, was created with MIT's NETLIST program. Leaf cells were then laid out using MAGIC, the new layout editor from Berkeley. Positioning of the leaf cells is performed using Coordinate Free LAP (CFL), a program developed at the UW/NW VLSI Consortium [2].

CFL was designed with generator creation in mind. As such it allows the designer to write a "C" program which embodies the general structure of the generator, even before the leaf cells have been designed. As these cells are created, or changed, CFL automatically aligns them in the correct fashion. CFL also provides the designer with autorouting primitives and, further, creates a complete description of the border of the newly-generated device. This enables manipulation of the device by any higher level program to proceed by accessing only a very small amount of data.

Once the layout phase was complete, the instances were extracted using MAGIC. Simulations with the switch-level simulator RNL were then performed to compare results with the transistor netlist, following which changes to NETLIST, the leaf cells and CFL were made to ensure agreement. Layout verification, and any further necessary model adjustments were then made prior to development of the test procedure. A self-test methodology based on BILBO[3] is currently being developed. This will be driven by the same input parameters as used by the CFL and NETLIST programs such that the result in a signature register (see Figure 1) can be predicted and used to detect simulated faults.

The complete data base representation of a generated microcontroller comprises four files, namely: a simulation file incorporating the transistor netlist, the mask information file containing the layout geometry information and design rule checker information from MAGIC, a border file giving the generator footprint from CFL, and a test file containing the test data.

A satisfactory degree of geometric rule independence is achieved by creating a master rule set from the rules of foundries likely to be employed in the microcontroller fabrication. The leaf cells are designed from this rule set using a graphical layout editor. This approach offers an alternative to the algorithmic technology file approach proposed elsewhere[4]. As our intended applications for microcontrollers are in the telecommunications field, both analog and digital circuitry are likely to be required on the chip. This limits the suitable fabrication technologies to those that use only single layer metal since "double-poly, double-metal" processes are not widely available.

III. DESIGN OBJECTIVES AND PRIORITIES

In establishing priorities for the design, the following axioms were considered to be appropriate to control systems in telecommunications: i) Control algorithms tend

†Funded in part by the Defense Advanced Research Projects Agency under Contract MDA903-85-K-0072.

to concentrate on bit manipulation operations rather than arithmetic operations; ii) the quantity of transient (i.e. read/write) data is relatively small and the data structures are simple; iii) control processors need not be high performance machines; iv) system timing is of critical importance in control applications.

Accordingly, it was determined that the priorities should be ordered as follows: size, simplicity and performance. To keep the design small, considerable effort was devoted to reducing the size of the largest elements (i.e. RAM, PLA, and ALU). To simplify the architecture, a reduced instruction set similar to RISC[5] was chosen. This was further simplified by treating all data as globals which reside in the same memory space, so eliminating the need for memory reference instructions. In addition, the number of control lines has been kept low to reduce bus routing problems. To maintain timing consistency, a constant cycle time for all instructions (including branches) was chosen. Performance issues were addressed in hardware by employing: (1) sub-generators with an algorithmic driver sizing capability[6]; (2) a modest pipeline and (3) separate data and control spaces. The resulting architecture is depicted in Figure 1.

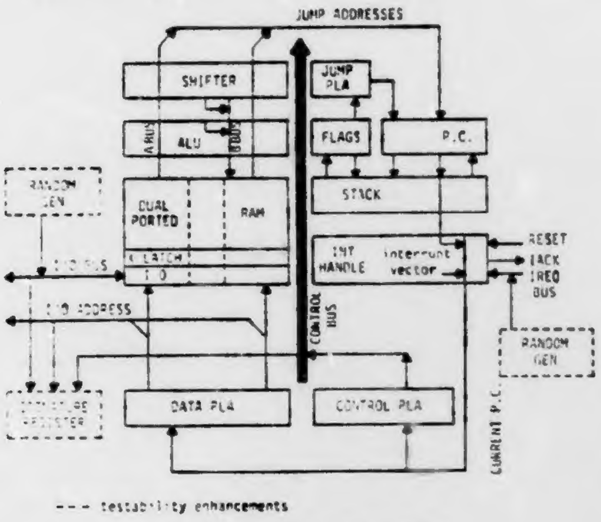

FIGURE 1. MICROCONTROLLER ARCHITECTURE

IV. ARCHITECTURE

Since both program and processor are contained within the same block, the system can be viewed as a finite state machine with the state feedback being the current program counter. A microcoding technique employing PLAs is used to produce the control and data path signals in parallel. Separation of these paths allows PLA minimization techniques to produce a more compact structure. The output of the PLAs remains valid for an entire instruction cycle, see Figure 2.

System timing is based on a two-phase non-overlapping clock scheme, with one machine cycle taking two ticks of each clock to complete. The machine cycle is partitioned into read and write half-cycles by a state clock derived from one of the system clock phases, see Figure 2. The RAM must be read in the first half-cycle and written-to in the second half-cycle. The output of the RAM is dynamically latched by C^2MOS latches, using NORA[7] circuit techniques. The RAM itself is implemented in domino CMOS.

The RAM output, consisting of the contents of registers, I/O data or immediates (which have passed through it) form the input to the ALU and shifter. These elements perform their required operation and force the result back to the register file on the Bbus only.

The RAM is optimized for speed and it is noteworthy that the slowest devices, namely the ALU (slow carry chain) and PLA (high capacitance on the term lines) are able to operate on cycle times which are one half that of the RAM. The RAM, ALU and shifter, which together form the data path, are pipelined so that one is precharging while the other is evaluating.

The data path output may consist of branch addresses which form the input to the program counter. A PLA is used to compare the flags with the instruction condition code. If a subroutine call is to be executed, the current program counter and flags are pushed on the stack.

Interrupts are controlled by the interrupt handler. When an interrupt is being requested; the associated vector is forced onto the current P.C.bus. This causes a jump to the state containing a call to the appropriate interrupt service routine.

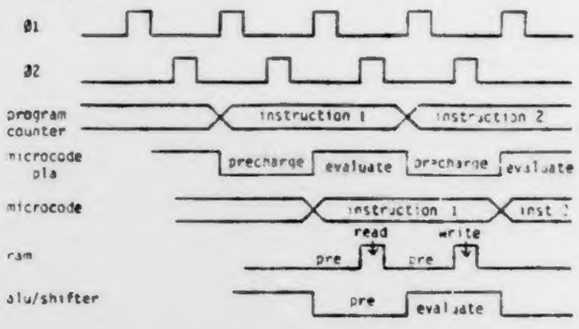

FIGURE 2. SYSTEM TIMING

V. GENERATOR INPUT PARAMETERS

The 9 lower level blocks of the controller are synthesized by 8 unique sub-generators. The input parameters to these generators are the register address width, number of interrupts, number of i/o ports, program counter width, stack depth and data path width. In addition, the ALU, shifter and branch PLA operations are defined. Many of these parameters can be synthesized by analysis of the instructions used in a particular program. This ensures a compact microcontroller which is capable of executing a local subset only of the global set available.

The ALU is an example of where this approach to area reduction is used. In addition to the data path width, the ALU generator has a switch which directs it to synthesize an "arithmetical/logical" or just a "logical" unit, depending on the particular operations required. The shifter generator works in a similar way and synthesizes only the actual paths used in the crossbar switch, instead of all those that are possible.

VI. INSTRUCTION SET

The global instruction set has been devised to provide a limited but powerful set of instructions. All instructions execute in one machine cycle and there are 22 instruction types, see Figure 3.

All of the read/write registers in the processor reside in the dual ported register file. The i/o address space is also memory mapped, therefore, there are no divisions between registers, memory and I/O ports. This greatly simplifies the instruction set while making it completely orthogonal. Several special (x) latches may also reside in the RAM. These allow indirect addresses to be stored and used to access the registers. Since the RAM is designed to have A and B busses read simultaneously but a write to the B bus only, all data path operations are of the form:

(Reg1/data/io) OP (Reg2/io) ---> (Reg2/io)

The source of jump or call addresses is the RAM as well, thus addresses may be immediate or computed. At present, 8 branch conditions are provided for.

By keeping the number of control lines small ($<=14$), it has proved possible to implement horizontal microcode (i.e. no instruction decode), producing in this case 784 unique instructions. This provides maximum flexibility while maintaining architectural simplicity. The result is that operations which do not use the data path (i.e. flags, interrupt control, return) can be executed in parallel with those that do.

DATA PATH OPERATIONS:

operand pairs:	operations:	
data, reg	ADD	SHLL
reg1, reg2	ADDC	SHLA
*x , reg	SUB	SHRL
reg , *x	SUBC	SHRA
data, *x	AND	RLL
x , reg	OR	RLA
reg , x	XOR	RRL
	MOVE	RRA

*x = an indirect address

OTHER OPERATIONS:

```
FLAGS      LATCH
CALL   ON  CONDITION
RETURN  ON  CONDITION
JUMP   ON  CONDITION
COMPARE
ENABLE/DISABLE  INTERRUPT
```

FIGURE 3. GLOBAL INSTRUCTION SET

VII. RESULTS

The generator has been designed in 3μ CMOS, using a conservative set of the MOSIS design rules. Several instances of the microcontroller have been simulated using NETLIST in conjunction with RNL. This has shown that the speed of a particular instance is highly dependent on the resources used. For example, for moderate word widths, the ALU carry chain delay dominates. If the carry chain is not present (i.e. logical operations only), the program counter becomes the speed determining factor. Most instantiations are expected to perform with a processing rate better than 2 million instructions/second.

Figure 4 is the check plot of a test instance with an 8 bit word width, 31 general purpose registers, 1 indirect address register and an 8 deep stack. Each subgenerator is fully implemented (i.e. all instruction types can be executed) and the program is 280 steps in length. The pad frame has a cavity which is 5.5mm on a side.

Some of the major data path items, notably the RAM and the shifter have already been fabricated. Initial testing shows that the functionality and performance track well with the expected results.

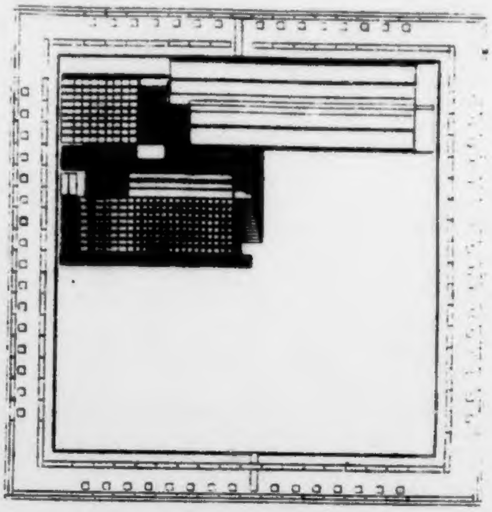

Figure 4. Check Plot

VIII. CONCLUSION

A design environment for the automatic generation of microcontrollers in single chip telecommunications applications has been developed. When the input parameters are supplied, the generator synthesizes the mask geometries for the required microcontroller. By analyzing the instructions used in a particular program, layouts of major components such as the ALU and the barrel shifter are defined. In this manner, a compact layout is produced for each different microcontroller instance. Several microcontrollers with data-paths ranging from 2 to 16 bits have been simulated. The corresponding machine cycle times are predicted to be 180 to 600 nanoseconds.

REFERENCES

1. Buric, M.R., C. Chistensen, and T. G. Matheson
"The Plex Project: VLSI Layouts of Microcomputers Generated by a Computer Program", Proceedings of IEEE International Conference on Computer Aided Design, Sept. 1983, pp. 49-50.

2. Beckett, W.
"Coordinate Free LAP Reference Manual -- Version 3.0", UW/NW VLSI Consortium Technical Manual.

3. Koenemann, B., J. Mucha and G. Zwiehoff
"Built-in Logic Block Observation Techniques", Digest 1979 Test Conference, 79CH1509-9C, October 1979, pp. 37-41.

4. Buric, M. R. and T. G. Matheson
"Silicon Compilation Environments", Proceedings IEEE Custom Integrated Circuits Conference, May 1985, pp. 208-12.

5. Katavenis, M. G. H.
"Reduced Instruction Set Computer Architectures for VLSI", Ph.D. Thesis, University of California at Berkeley, Oct. 1983.

6. Jinks, B. W. , W. S. Snyder and D. L. Pulfrey,
"The Algorithic Generation of ROM Macrocells", Proceedings of the Second International Symposium on VLSI Technology, Systems and Applications, May 1985.

7. Goncalves, N. F. and H. J. DeMan,
"NORA: A Racefree Dynamic CMOS Technology for Pipelined Logic Structures", IEEE Journal of Solid State Circuits, Vol. SC-18, No. 3, June 1983, pp.261-266.